OpenCV Face Recognition App Documentation

In this project, we utilize the "assets" folder to store images and fonts. The folder path is **"/opencv-fr-mobile/assets".**

Let's explore the code structure and functions used in the following directories:

- **lib**: This directory contains the main Dart code files for the project.
- **android**: Contains the Android-specific files and configurations.
- **ios**: Contains the iOS-specific files and configurations.
- **test**: Contains test files for unit and integration testing.
- **assets**: Contains images and fonts used in the project. The path for this folder is "/opencv-fr-mobile/assets".

Let's further examine each directory's contents to understand the functionality and features of our Face Recognition API.

In this project, we utilized the following plugins.

- **cupertino_icons**: This package provides the Cupertino icons for iOS-style design. It's commonly used in Flutter projects, especially if you're developing for iOS.
- **http**: This package is used to make HTTP requests in Flutter apps. It's essential for fetching data from APIs or making any kind of network requests.
- **image**: This package provides functionality for working with images in Flutter, like decoding, encoding, and manipulating images.
- **flutter_localizations**: This package provides localizations for many languages, which is essential if your app needs to support multiple languages.
- **intl**: The intl package provides internationalization and localization support. It's commonly used for formatting dates, numbers, currencies, and strings according to different locales.
- **flutter_native_splash**: This package is used to customize and generate native splash screens for both Android and iOS.
- **cached_network_image**: It provides caching functionality for network images in Flutter. This improves performance by caching images locally, reducing unnecessary network requests.

- **flutter_svg**: This package allows you to display SVG (Scalable Vector Graphics) files in Flutter applications. SVG is a vector image format, which means it scales well without losing quality.
- **flutter_screenutil**: This package provides a convenient way to size widgets dynamically based on the device's screen size and pixel density.
- **path_provider**: This package provides a platform-agnostic way to access the filesystem paths for storing and retrieving files.
- **camera** and **image_picker**: These packages are used for accessing the device's camera and image picker functionality, respectively. They are essential for capturing photos or selecting images from the device's gallery.
- **bloc** and **flutter_bloc**: These packages are used for state management using the BLoC (Business Logic Component) pattern. They help in separating business logic from UI components and managing state in a predictable way.
- **equatable**: Equatable is used in conjunction with BLoC for easier comparison of objects in Dart. It overrides == and **hashCode** methods to compare object values.
- **flutter_statusbarcolor_ns**: This package provides methods to change the status bar color on Android and iOS. It's useful for customizing the status bar color in Flutter apps.
- **dotted_border**: This package provides a widget to draw dotted borders around other widgets.
- **pinput**: This package provides customizable PIN input fields for Flutter apps.
- **shared_preferences**: SharedPreferences allows you to store simple data locally on the device. It's commonly used for storing user preferences or small amounts of app data.
- **firebase_messaging** and **flutter_local_notifications**: These packages are used for integrating push notifications into Flutter apps using Firebase Cloud Messaging (FCM).
- **firebase_core**: This package is required for initializing Firebase in Flutter apps.
- **url_launcher**: This package provides a way to open URLs in the device's browser. It's used for launching external links from the app.
- **synchronized**: This package provides synchronized objects for Dart. It's useful for ensuring thread safety when working with asynchronous code.
- **page_transition**: This package provides custom page transition animations for Flutter apps.
- **fluttertoast**: Fluttertoast is used for displaying toast messages in Flutter apps.
- **flutter_keyboard_visibility**: This package provides methods to detect whether the keyboard is visible on the screen.
- **flutter_image_compress**: This package allows you to compress images in Flutter apps, which is useful for reducing file sizes before uploading or storing images.
- **recaptcha_enterprise_flutter**: This package integrates Google's reCAPTCHA Enterprise service into Flutter apps for verifying human users.

- **package_info_plus**: This package provides information about the application package, like version, build number, etc.
- **phone_numbers_parser**: It's used for parsing phone numbers and extracting information like country code, national number, etc.
- **country_code_picker**: This package provides a country code picker widget for Flutter apps. It's used to select a country code from a list of countries.
- **baseflow_plugin_template**: This seems to be a template plugin package. It might be used for creating custom plugins for Flutter apps.
- **permission_handler**: This package provides methods to request and check permissions on both Android and iOS.
- **connectivity_plus**: Connectivity Plus is used for monitoring network connectivity in Flutter apps. It provides information about the device's network status.
- **multi_dropdown**: This package provides a widget for selecting multiple items from a dropdown list.
- **flutter_awesome_select**: It provides a customizable dropdown widget with support for multiple selection and filtering.
- **sticky_headers**: This package provides sticky headers for lists in Flutter. It's used to make headers stick to the top of a list when scrolling.
- **tutorial_coach_mark**: This package is used for creating and displaying coach marks or tutorials in Flutter apps. It's useful for guiding users through app features or UI elements.

In the **opencv-fr-mobile** project, the Firebase integration includes the following files:

- **GoogleService-Info.plist**: Located at **opencv-fr-mobile/ios/Runner/GoogleService-Info.plist**, this file is used for iOS.
- **google-services.json**: Located at **opencv-fr-mobile/android/app/google-services.json**, this file is used for Android.
- **recaptcha_options.dart**: This file contains the reCAPTCHA key for both Android and iOS.
- **firebase_options.dart**: This file contains Firebase options details.

The project structure includes a **lib** folder (**opencv-fr-mobile/lib**) where the code for the entire system resides. Within this **lib** folder, we follow the bloc structure to manage state and separate business logic from UI components. These files handle various functionalities such as push notification management, onboarding, login, and home screen redirection.

Inside the **lib** folder, you can find three main folders: **config**, **core**, and **utils**. The **config** and **core** folders are used for common activities, functionality, custom UI design, and base design.

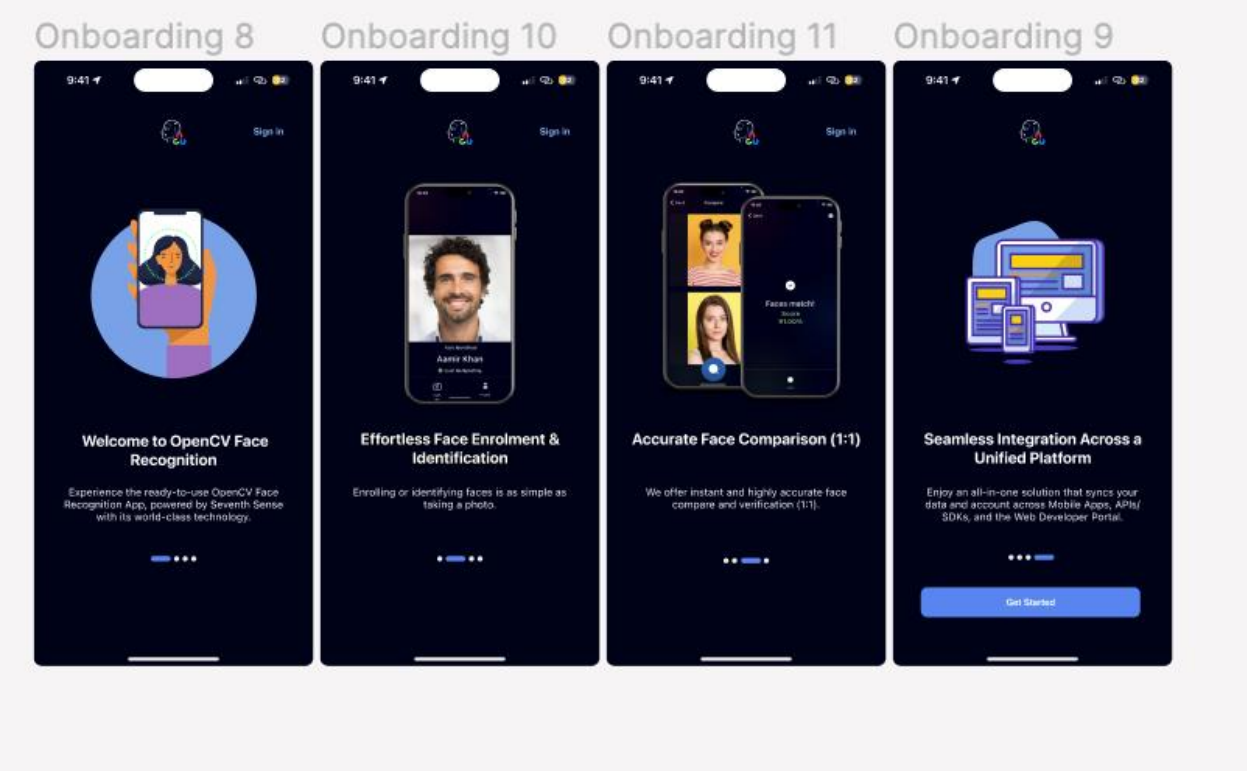
- **lib -> config -> l10n -> app_en.arb**: This file contains all the localization strings.
- **lib -> config -> theme -> data -> app_theme.dart**: This file contains the app theme settings, including dark mode and light mode.

In the **core** folder, we have sub-folders for various constants, networking, services, utils, and widgets:

- **lib -> core -> constants**:
    - **api_constants.dart**: Manages the API endpoints for both production and development.
    - **app_paddings.dart**: Manages the gaps between two objects.
    - **color_codes.dart**: Manages all the color codes.
    - **email_validate.dart**: Manages email validation.
    - **error_codes.dart**: Manages all the error codes.
    - **error_messages.dart**: Manages custom error messages.
    - **font_family.dart**: Manages custom font families.
    - **nationalities.dart**: Manages all the nationalities.
- **lib -> core -> networking**:
    - **api_model.dart**: Manages the API models like the base API response class.
    - **api_exception.dart**: The base class for all API exceptions.
    - **api_provider.dart**: The base class for all API endpoints.
- **lib -> core -> services**: Inside this folder, we have several classes for storing data, setting the base URL, handling reCAPTCHA services, and detecting the platform currently in use (Android or iOS).
- **lib -> core -> utils**: Inside the **utils** folder, we handle image size calculation, image conversion, image picking from the gallery, and asset declaration.
- **lib -> core -> widgets**: Inside the **widgets** folder, we maintain custom common UI classes, such as common alerts, custom buttons, bottom snack bars, text fields, common app navigation bars, common back buttons, and common vertical and horizontal gaps.

Onboarding:

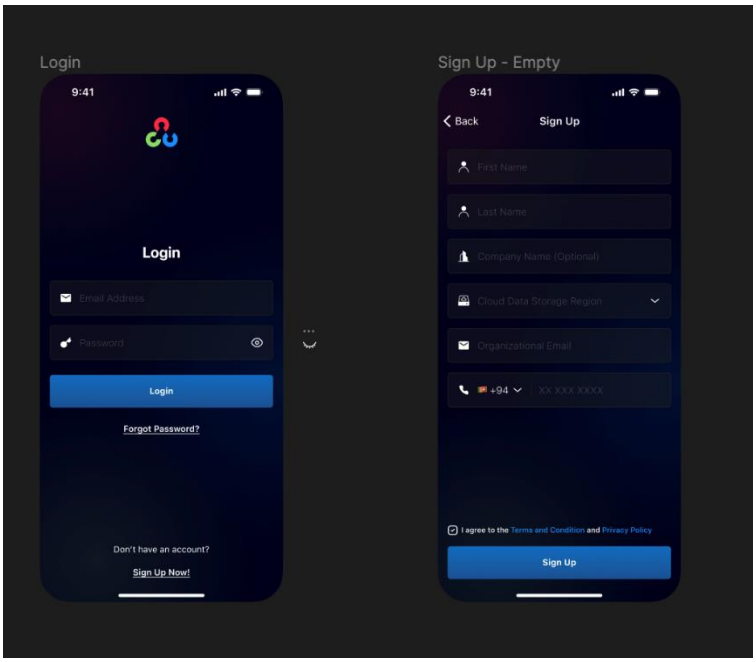
In the **features/onboarding** directory, we develop the UI screens and their functionalities.

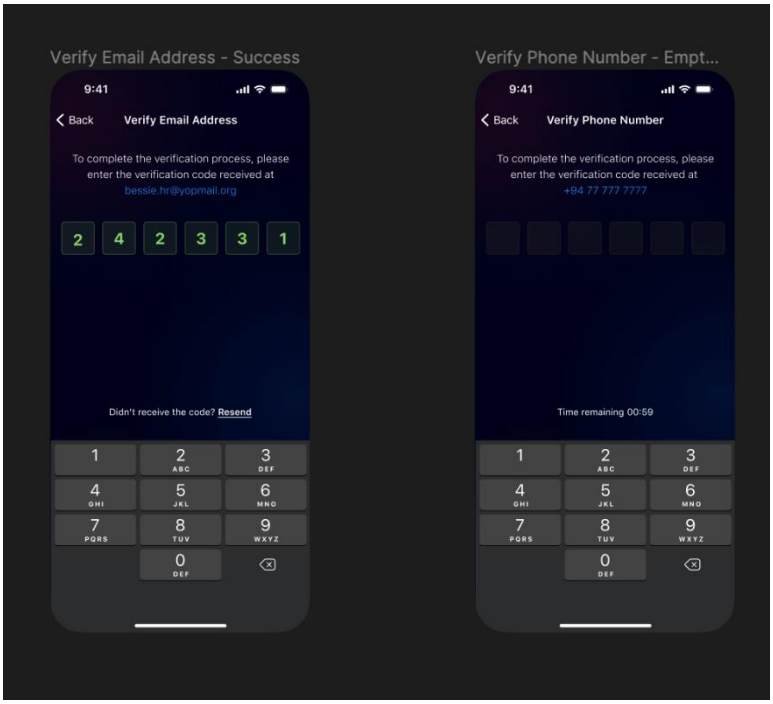


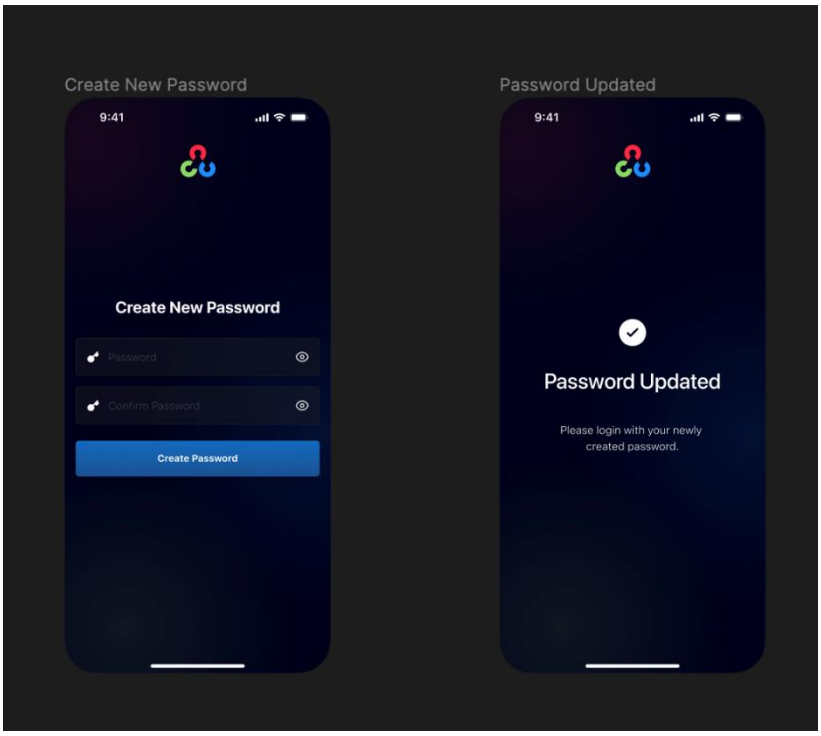
Authentication:

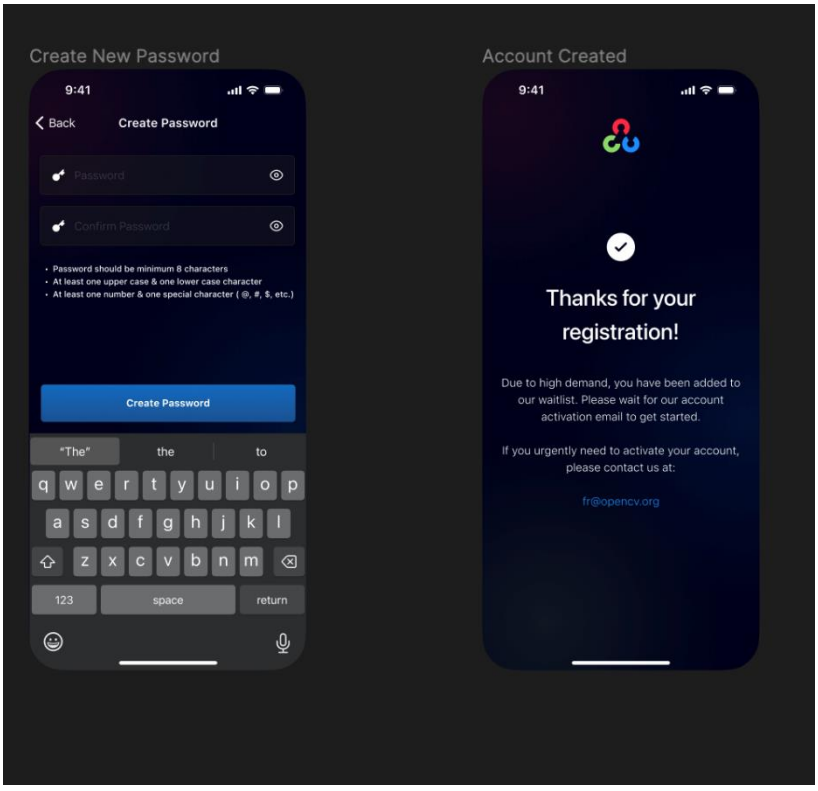
Under **features/authentication/presentation/widgets**, we create custom UI designs like back buttons, dropdowns, phone number code pickers, OTP codes, and text fields. In **features/authentication/presentation/screens**, we write the UI code for authentication screens,

and their actions are handled by **RegisterBloc** and **AuthBloc** classes. API endpoints are called from the **Data/repository** folder.



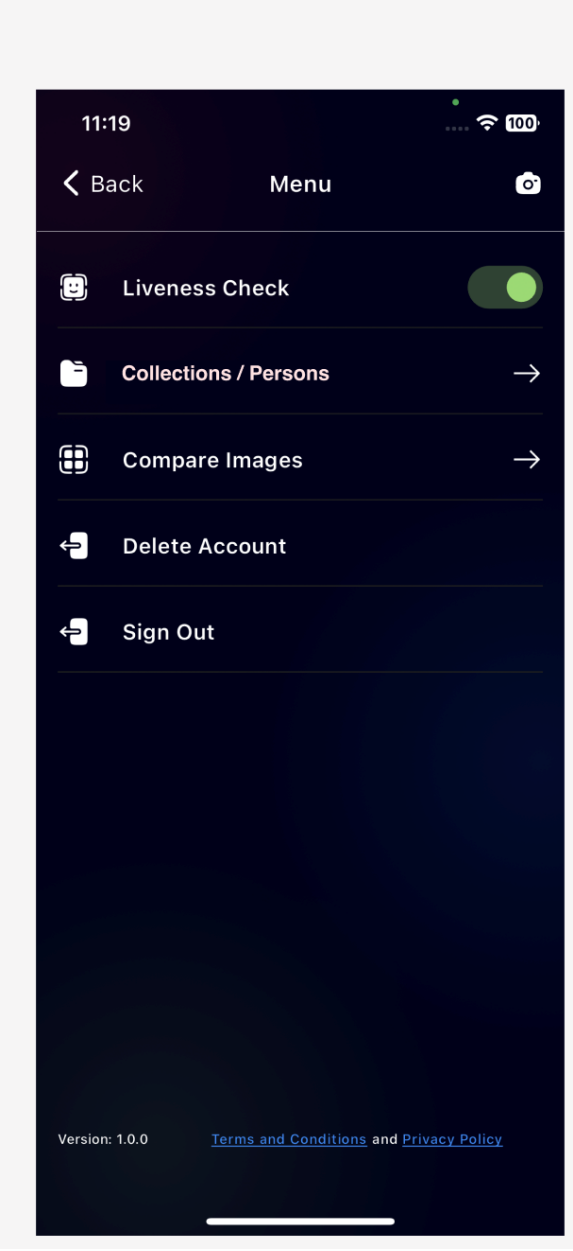



Create New Password
9:41
Create New Password
Password
Confirm Password
Create Password
Password Updated
9:41
Password Updated
Please login with your newly created password.


Create New Password
9:41
Back
Create Password
Password
Confirm Password
• Password should be minimum 8 characters
• At least one upper case & one lower case character
• At least one number & one special character (@, #, $, etc.)
Create Password
"The"
the
to
Account Created
9:41
Thanks for your registration!
Due to high demand, you have been added to our waitlist. Please wait for our account activation email to get started.
If you urgently need to activate your account, please contact us at:
fr@opencv.org

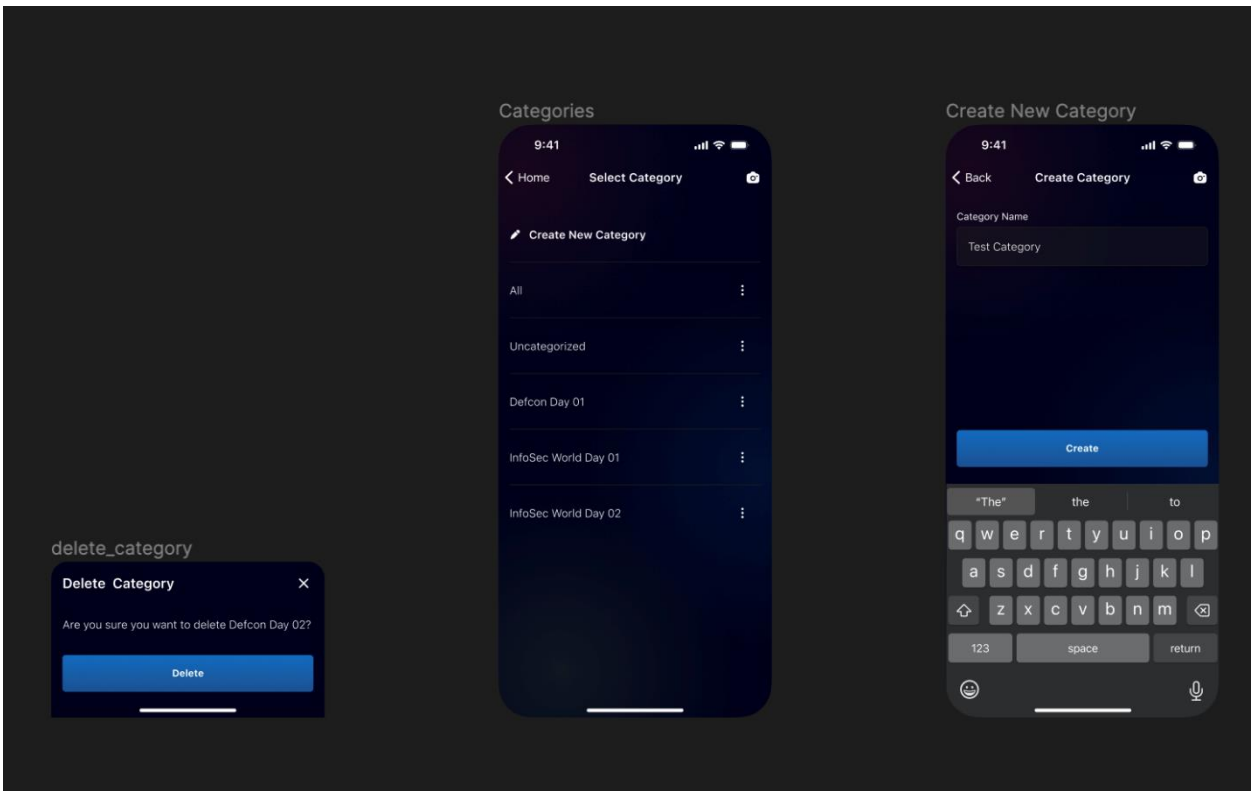
Side Menu:

In **features/sidemenu**, we implement open menu navigation to access the side menu and its functionalities. We use **MenuBloc** to fetch user details, control liveness, and manage account deletion. The logout action is handled by **AuthBloc**.



Collection:

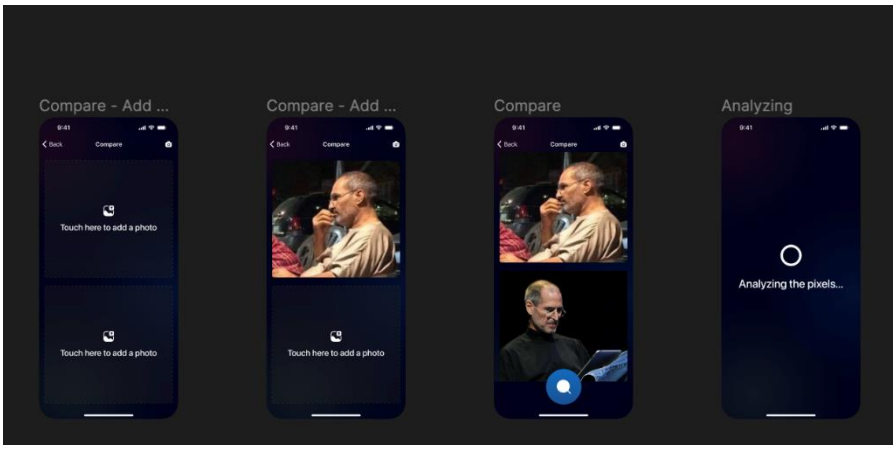
Clicking on the side menu's "Person" row opens the collection details. UI screens are located in the **presentation** folder, and actions like delete, edit, and create are managed by the bloc.

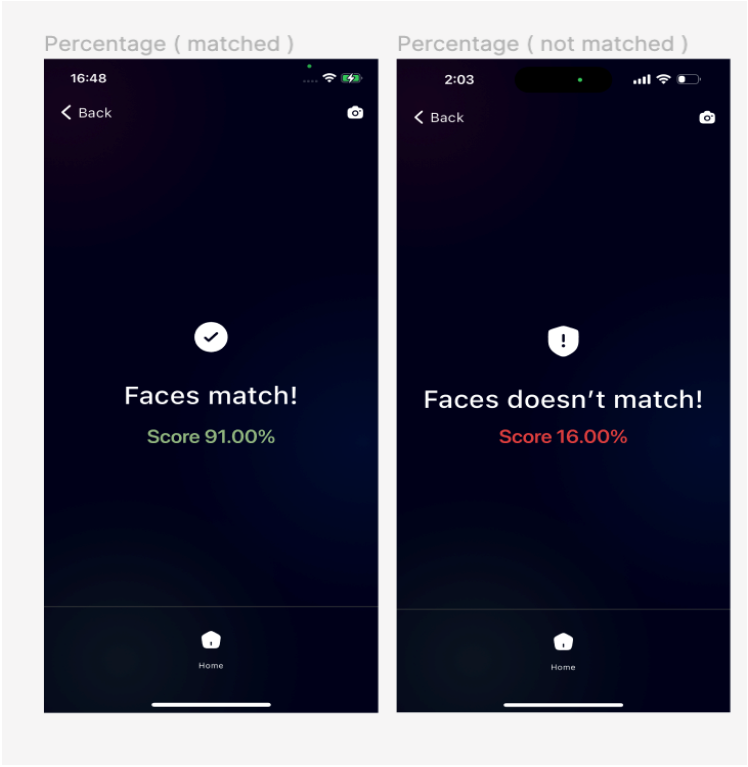


Compare Photos:

In **features/compare_photos**, we compare two images. UI screens are under the **view** folder,

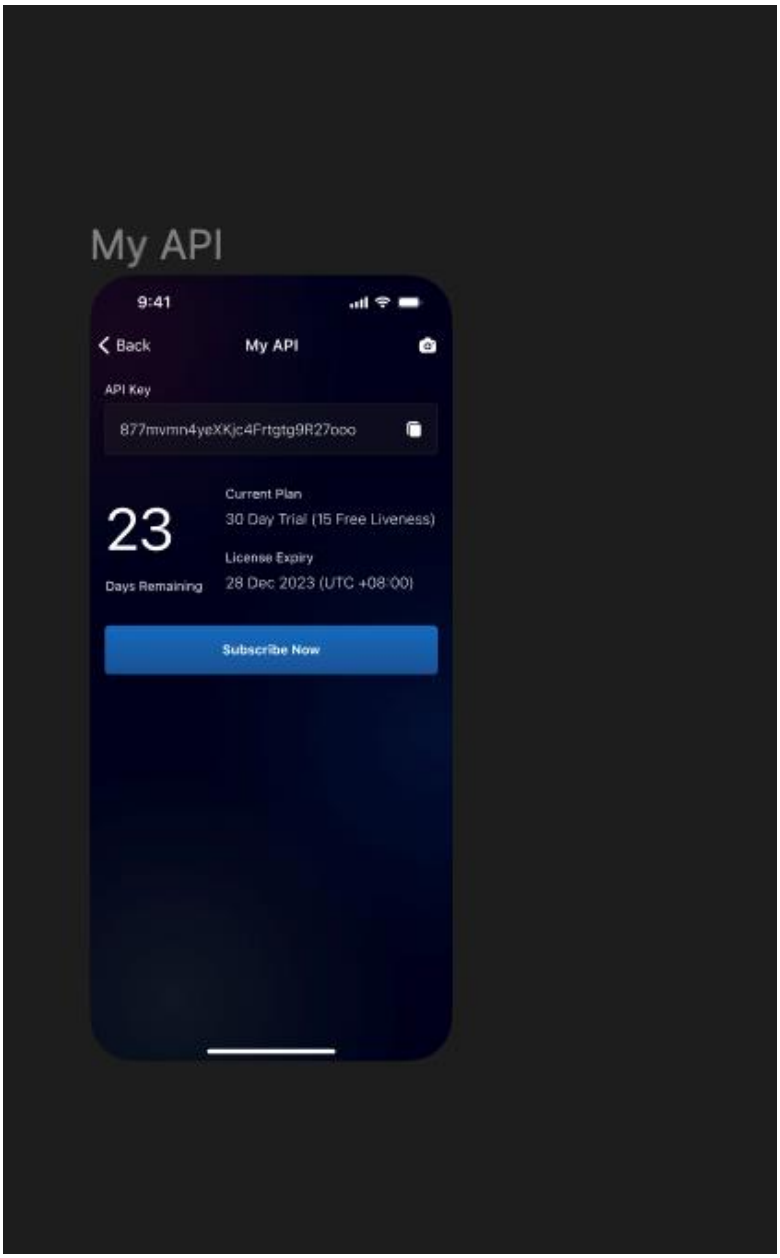
and search button actions are handled by the bloc, with API endpoints called from the repository folder.





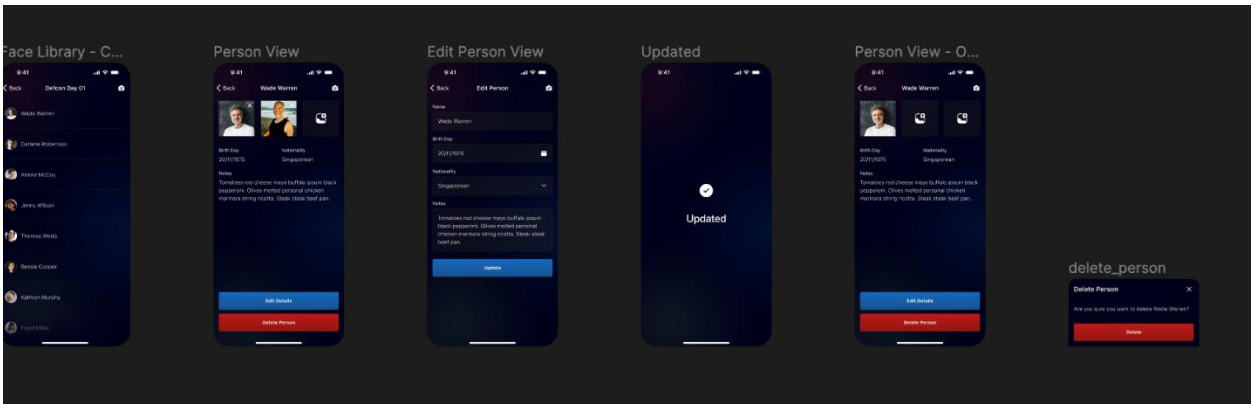
My API:

In **features/** myapi, This folder provides user details. UI screens and their functions are located in the **presentation** folder. Fetching user details is handled by the bloc, and API endpoints are called from the subscription repository file.



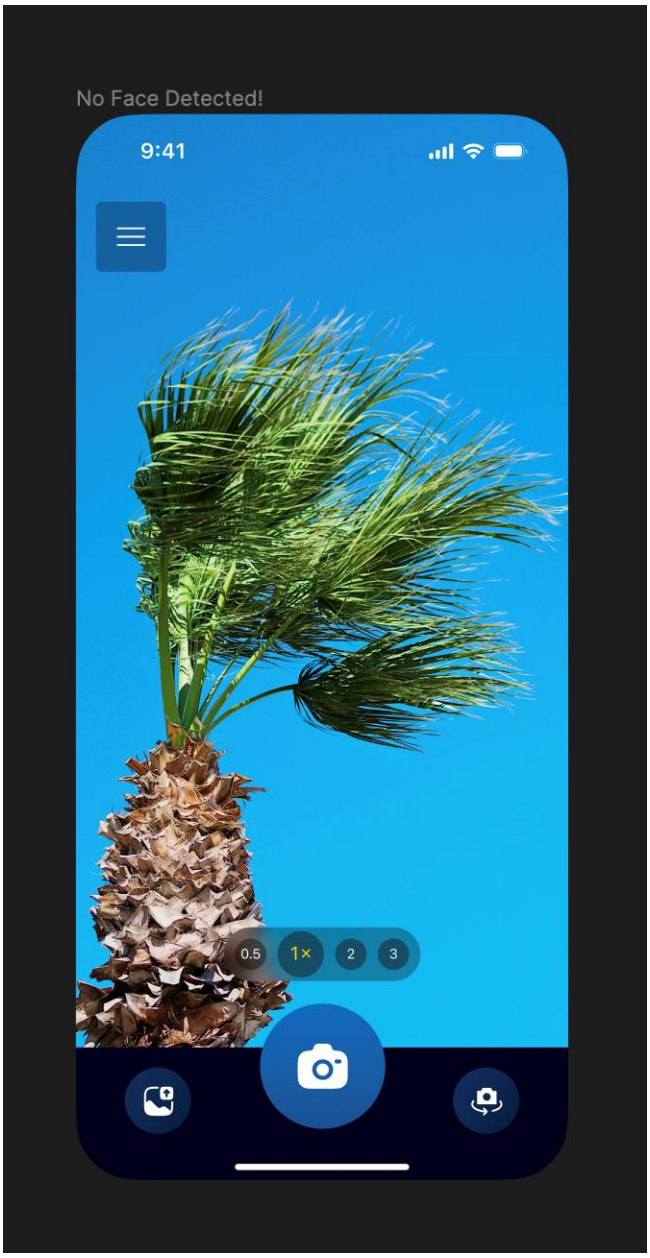
Person:

In **features/**person, Functions and actions related to user details are managed in this folder. UI screens and functions are in the **presentation** folder. Fetching person details is handled by the bloc, and API endpoints are called from the person repository file.



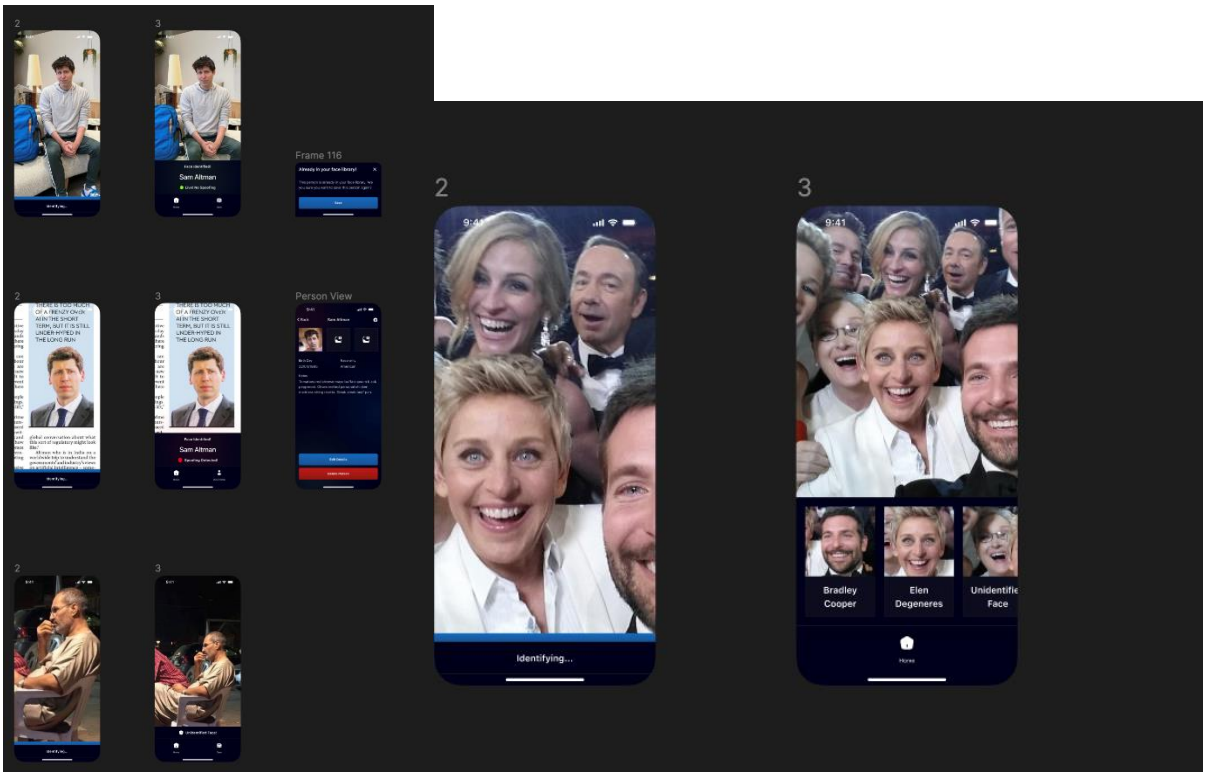
Dashboard:

UI functionalities are developed in this folder. Inside **features/dashboard/presentation**, we design UI screens and functions. The bloc manages camera functions such as toggling, taking photos, and fetching images from the gallery.



Search:

UI screens and actions for search functionality are managed here. Inside **features/search/page**, we design UI screens and functions. The bloc handles calling the search API and actions such as saving and viewing persons.


2
3
Sam Altman
Frame 116
Already in your face library!
Person View
Sam Altman
2
3
Identifying...
Bradley Cooper
Elen Degeneres
Face
Home